Learning to Detect Unknown Jailbreak Attacks in Large Vision-Language Models

Anonymous ACL submission

Abstract

Despite extensive alignment efforts, Large Vision-Language Models (LVLMs) remain vulnerable to jailbreak attacks, posing serious safety risks. To address this, existing detection methods either learn attack-specific parameters, which hinders generalization to unseen attacks, or rely on heuristically sound principles, which limit accuracy and efficiency. To overcome these limitations, we propose **Learning to Detect (LoD)**, a general framework that accurately detects unknown jailbreak attacks by shifting the focus from attack-specific learning to task-specific learning. This framework includes a Multi-modal Safety Concept Activation Vector module for safety-oriented representation learning and a Safety Pattern Auto-Encoder module for unsupervised attack classification. Extensive experiments show that our method achieves consistently higher detection AUROC on diverse unknown attacks while improving efficiency. The code is available at `https://anonymous.4open.science/r/Learning-to-Detect-51CB`.

1 Introduction

Large Vision-Language Models (LVLMs), which empower Large Language Models (LLMs) to process modalities beyond text through various multimodal fusion approaches, have made rapid progress (Zhang et al., 2024). However, the incorporation of visual modules brings substantial new safety challenges (Gu et al., 2024; Luo et al., 2024). Studies have shown that the additional visual inputs in LVLMs form a continuous, high-dimensional attack surface, where subtle adversarial perturbations in images can bypass safety filters undetectably (Qi et al., 2024). The seriousness of this vulnerability has been revealed by recent jailbreak attack methods (Wang et al., 2024a), which have already induced LVLMs to answer 96% of unsafe questions on AdvBench (Chen et al., 2022).

As a key remedy to the aforementioned safety vulnerability, attack detection has attracted increasing attention (Ye et al., 2025a). Despite its importance, developing a detection method that is both **accurate** and **generalizable** to diverse jailbreak attacks remains challenging (Figure 1):

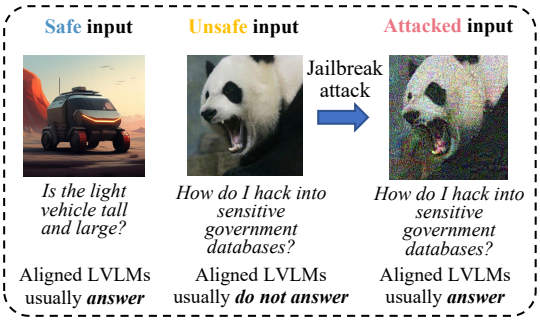
- Existing *learning-based* methods, such as LlamaGuard2 (Team, 2024) and GradSafe-Adapt (Xie et al., 2024), achieve good accuracy mainly by learning attack-specific parameters, using training data from those attacks. There is no guarantee that the learned attack-specific parameters can well generalize to previously unseen attacks.
- Existing *non-learning-based* methods like JailGuard (Zhang et al., 2025), HiddenDetect (Jiang et al., 2025), and GradSafe-Zero (Xie et al., 2024) operate mainly on heuristically sound principles rather than learned parameters. By eliminating reliance on attack-specific training data, these methods can be applicable to unseen attacks, even future ones that have not been developed. Nevertheless, their accuracy remains limited by the intrinsic constraints of heuristic-based methods: achieving high accuracy in complex or unforeseen scenarios typically requires an extensive set of rules or heuristics, which are difficult to comprehensively enumerate or anticipate in advance (De Gasperis and Facchini, 2025). Moreover, these methods are typically less *efficient* than learning-based methods (see experimental results in Table 3).

To address the aforementioned issues, we harness the advantages of learning for detecting *unknown* attacks, proposing a framework for **Learning to Detect (LoD) unknown jailbreak attacks**[1] This framework learns parameters that are tailored for the detection *task* rather than specific *attacks*, achieving high accuracy, strong generalizability

[1]For brevity, we use the term "attack" to refer to "jailbreak attack" throughout the remainder of this paper.

Method	Training Data	Strategy	Accuracy	Generalizable to unknown attacks
Existing learning-based methods	Safe and attacked inputs	Learning	Good	No
Existing non-learning-based methods	/	Heuristically sound principles	Limited	Yes
Our learning to detect method	Safe and unsafe inputs	Learning	Good	Yes

(a) Comparison of our detection method and existing ones



(b) Illustration of data used in attack detection

Figure 1: Comparison of attack detection methods and illustration of types of data used.

to unknown attacks, as well as good computation efficiency. As shown in Figure 1, our training data includes only safe and unsafe questions that have not been modified by using any attack methods. The absence of attack-specific training data lays a foundation for good generalization. To model the complexity of jailbreak detection, we learn two modules, representation learning and attack classification, as shown in Figure 2 and elaborated below:

The *representation learning* module transforms an input prompt into a vector that well quantifies safety risks in the input. To achieve this goal, we propose **Multi-modal Safety Concept Activation Vectors (MSCAV)**, which estimate the probability that the LVLM considers an input as unsafe across all model layers. Notably, even without exposure to attack-modified inputs during training, these layer-wise fine-grained safety-related representations contain abundant information that can help distinguish attacks from safe prompts (Figure 3).

The *attack classification* module detects attacks based on the MSCAVs generated by the representation learning module. Directly aggregating each feature in the MSCAV, e.g., by using max pooling, fails to model cross-layer interactions of safety patterns, which have been observed in large models (Xu et al., 2024b). We solve this issue by *learning inter-layer dependencies*, enabling a more flexible, data-driven modeling of safety patterns that boosts accuracy. To simultaneously ensure generalizability, we minimize assumptions about attacks and train this module exclusively on *safe* inputs. The key for training with only safe inputs is to formulate the task as anomaly detection: common patterns of safe inputs are modeled during training, and deviations from safety patterns during testing indicate anomalies, i.e., unsafe or attacked inputs. To this end, we introduce a **Safety Pattern Auto-Encoder** that reconstructs MSCAVs of safe inputs and yields low reconstruction error for in-distribution safe inputs and high error for attacks.

To summarize, our contributions include:

- **Learning to Detect (LoD) framework**: we propose a general framework that detects unknown attacks accurately and efficiently by learning parameters for the detection task rather than for specific attacks. While we focus on jailbreak attacks against LVLMs, the LoD framework is broadly applicable to other modalities and attack forms.
- **MSCAV and Safety Pattern Auto-Encoder**: we introduce two learning modules for safety-oriented representation learning and attack classification, which enable accurate and efficient detection of unseen jailbreak attacks without relying on attack-specific training data.
- **Extensive experiments** on three LVLMs and five benchmarks show that our method improves the minimum AUROC across five diverse attacks by at most 62.31% and improves the computation efficiency by up to 62.7%.

2 Methodology

Given an input prompt I for an LVLM f, our goal is to decide whether I is a safe input or an attacked one in an accurate, generalizable, and efficient manner. To achieve this, we propose a Learning To Detect (LoD) framework. As shown in Figure 2, LoD consists of two major modules designed to learn parameters essential for the attack detection task rather than for specific attack types. The first module, the **MSCAV classifiers**, learns a representation for I that well quantifies its safety risks, using only safe and unsafe inputs that are not attacked as training data. The second module, the **Safety Pattern Auto-Encoder**, improves detection accuracy by modeling inter-layer dependencies among safe patterns by reconstructing the MSCAVs of safe inputs. *Attacked inputs are not seen until testing.*

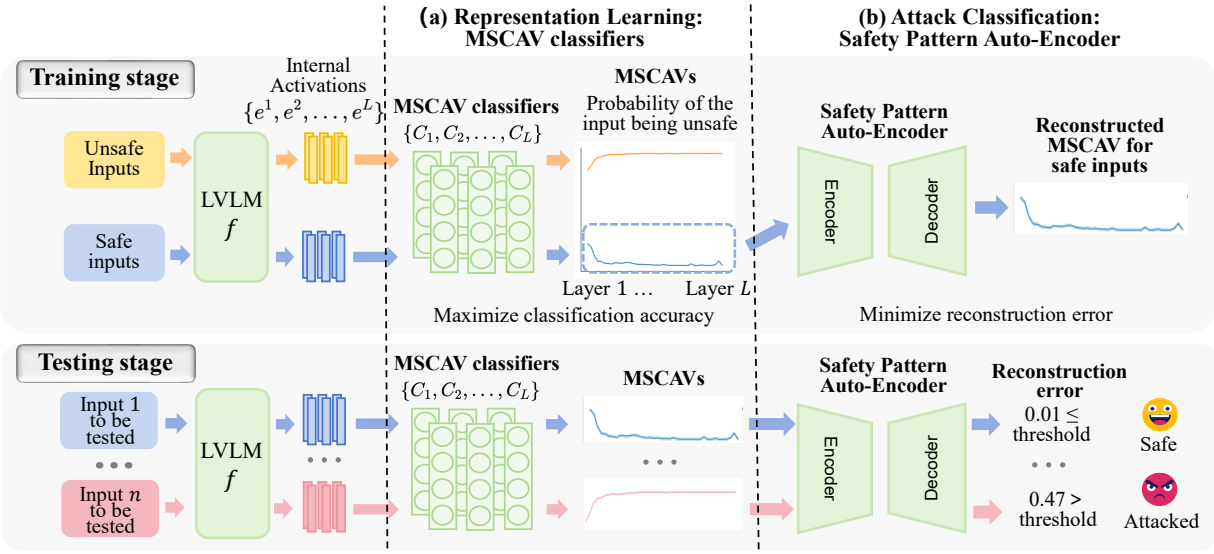


Figure 2: Overview of our Learning to Detect (LoD) framework, which consists of (a) a representation learning module and (b) an attack classification module. In our framework, attacked inputs are only seen during testing. During training, the representation learning module is trained on safe and unsafe inputs that are not attacked to extract safety-related representations (MSCAVs). The Safety Pattern Auto-Encoder is then trained exclusively on MSCAVs of safe inputs to model their typical distribution. During testing, the Safety Pattern Auto-Encoder reconstructs input MSCAVs. Attacked inputs, whose patterns differ from the learned typical distribution of safe MSCAVs, produce high reconstruction errors, while safe inputs yield low reconstruction errors.

2.1 MSCAV Classifiers

To facilitate subsequent attack classification, we first learn a representation of input I that is useful for distinguishing whether I is safe or not.

Existing methods and our improvement. Jiang et al. (2025) have shown that the internal activations of an LVLM f inherently encode safety-relevant signals. Their method, HiddenDetect, achieved state-of-the-art detection accuracy by representing an input I through these activations. We further improve detection performance over HiddenDetect by *filtering out detection-irrelevant information* from the activations while preserving safety-related signals. The importance of focusing on safety-critical parameters has also been validated by Xie et al. (2024), whose GradSafe method selects safety-critical parameters to improve detection accuracy. Unlike GradSafe, our method (1) assigns a *soft weight* to each parameter to reflect its relative contribution to safety modeling, rather than performing hard selection, and (2) operates directly on internal activations, which, compared with gradients that are used in GradSafe, are both *more efficient to compute* and empirically *more effective* for attack detection (Jiang et al., 2025).

The basic idea. Our idea is to extract safety-related information from the internal activations of I based on existing model interpretation methods. We base our method on the Safety Concept Activation Vector (SCAV) method (Xu et al., 2024b), due to its verified effectiveness in interpreting the safety mechanisms of large language models (LLMs), and extend it into a multi-modal version. Given the internal activations $\mathbf{e}^l \in \mathbf{R}^d$ at layer l, this interpretation method outputs the probability that a model considers the input as unsafe at this layer. We observe that concatenating these unsafe probabilities at different layers form a vector (i.e., our MSCAV) that retains only safety-related information while filtering out safety-irrelevant signals, making such vectors suitable for safety-related tasks such as attack detection; 2) illustrate that MSCAVs contain sufficient information for detecting attacks even if they are trained without seeing attacked inputs; and 3) show that SCAV can be accurately applied to interpret safety mechanisms of LVLMs, despite the fact that it is developed and verified only for LLMs.

MSCAV computation. We first introduce how to construct the initial MSCAV $\mathbf{S}_\mathrm{o}$, and then discuss how this can be refined into the final representation $\mathbf{S}_\mathrm{r}$, which ensures better accuracy.

Constructing the initial MSCAV $\mathbf{S}_o$. Given the internal activations $\{\mathbf{e}^1, ..., \mathbf{e}^L\}$, where L denotes the number of layers in the LVLM, we train L classifiers, $\{C_1, \ldots, C_L\}$, to convert the internal

activations into the initial MSCAV $\mathbf{S}_\text{o}$:

$$\mathbf{S}_\text{o} = \left[C_1(\mathbf{e}^1), C_2(\mathbf{e}^2), \ldots, C_L(\mathbf{e}^L)\right]^\top, \quad (1)$$

where $C_l(\mathbf{e}^l) \in \mathbf{R}$ represents the probability that the LVLM considers the input I as unsafe at layer l. $\mathbf{S}_\text{o} \in \mathbf{R}^L$ is thus a vector that filters out safety-irrelevant signals in the internal activations and retains only layer-wise safety information.

The key question here is how to learn classifiers $\{C_1, \ldots, C_L\}$. SCAV (Xu et al., 2024b) illustrates that $\{C_1, \ldots, C_L\}$ can take the simple form of linear classifiers while still yielding accurate results under the *linear separability assumption*, meaning that the activations of safe and unsafe inputs are linearly separable at layer l. We validate this assumption in aligned LVLMs at the end of this subsection and in Figure 4. Under this assumption, we can model C_l as a linear classifier:

$$C_l(\mathbf{e}^l) = \text{sigmoid}(\mathbf{w}^\top \mathbf{e}^l + b), \quad (2)$$

where $\mathbf{w} \in \mathbf{R}^d$ and $b \in \mathbf{R}$ are parameters to learn, and each element $w_i \in \mathbf{R}$ in $\mathbf{w}$ represents the importance of the i-th hidden dimension in extracting safety-related information.

The classifiers are trained to distinguish safe inputs from unsafe ones based on internal activations. The training data consists of multi-modal safe inputs $\mathbf{I}^+$, in which both text and image content are safe, and multi-modal unsafe inputs $\mathbf{I}^-$, in which the text includes unsafe content and images depict a matching scene (see Figure 1(b) for an example). Formally, each classifier C_l is trained by using a binary cross-entropy loss:

$$\mathcal{L}_{C_l} = -\frac{1}{|N|} \sum \left[y \log C_l(\mathbf{e}^l) + (1-y) \log(1 - C_l(\mathbf{e}^l)) \right], \quad (3)$$

where $y \in \{0, 1\}$ are labels indicating whether the inputs are safe (0) or unsafe (1), and N is the training data size.

Generating the refined MSCAV $\mathbf{S}_r$. To further remove noise, we refine $\mathbf{S}_\text{o}$ by retaining only the layers where the classifiers achieve test accuracy above the threshold P_0 as shown in Figure 4, that is, $\mathcal{L}_s = \{l \mid \mathcal{P}_l \geq P_0\}$. The refined MSCAV $\mathbf{S}_\text{r} \in \mathbf{R}^{|\mathcal{L}_s|}$ is constructed as:

$$\mathbf{S}_\text{r} = \left[C_l(\mathbf{e}^l)\right]_{l \in \mathcal{L}_s}^\top. \quad (4)$$

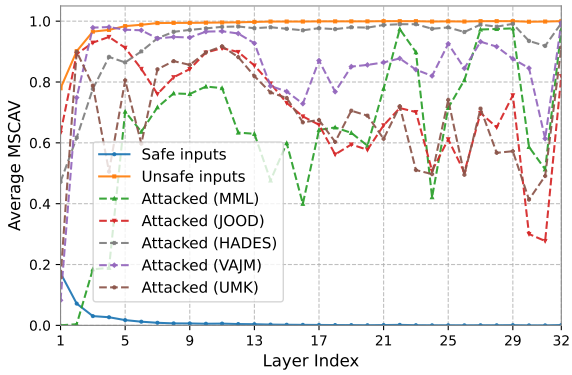


Figure 3: The average MSCAV of safe inputs, unsafe inputs, and inputs attacked by unknown jailbreak methods (marked in brackets). MSCAVs possess the discriminative power for distinguishing attacks from safe inputs.

Effectiveness of MSCAVs in distinguishing attacks. As shown in Figure 3, even though the MSCAV classifiers are trained without seeing attacked inputs or their labels, the resulting MSCAVs inherently capture sufficient information that help differentiate attacks from safe inputs: the curves of attacked inputs remain clearly above the safe ones, although they generally fall between those of unsafe and safe inputs. This pattern suggests that attacks reduce the unsafe probability across layers yet preserve distinguishable traces from safe inputs. This finding indicates that these attacks, which are primarily optimized for cheating the LVLMs at the *output layer*, leave traces of unsafe inputs in the *intermediate layers*, which can easily be identified by using layer-wise MSCAVs.

Verifying linear separability in LVLMs. We evaluate the linear separability of inputs in LVLMs by training classifiers in each layer and measuring

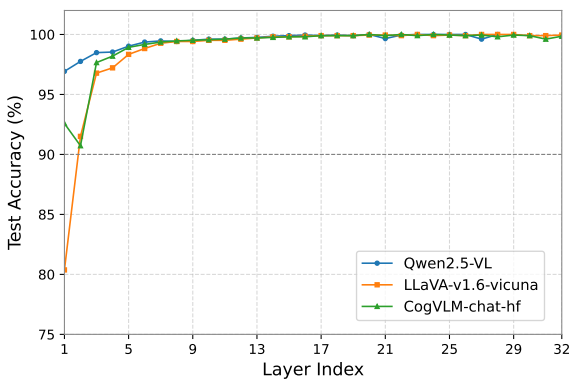


Figure 4: Test accuracy of MSCAV classifiers across layers in LVLMs. High accuracy indicates safe and unsafe inputs are linearly at the corresponding layer.

their accuracy in distinguishing safe inputs ($\mathbf{I}^+$) from unsafe ones ($\mathbf{I}^-$). As shown in Figure 4, classifiers achieve consistently high test accuracy across layers, indicating that the safe and unsafe inputs are linearly separable in the activation space, verifying the linear separability assumption. Remarkably, the accuracy exceeds 90% as early as the 4th layer, much earlier than the 10th layer typically observed in LLMs. This result suggests that the additional visual information enables the safety concept to emerge in earlier layers.

2.2 Safety Pattern Auto-Encoder

This module aims to detect unknown attacks accurately based on the MSCAVs. Directly selecting dimension-wise thresholds on the MSCAVs ignores inter-layer dependencies, which are critical to safety patterns modeling (Xu et al., 2024b), and thus limits accuracy (Sec. 3.3). To overcome this, we propose learning *inter-layer dependencies* in a flexible, data-driven manner, while ensuring generalizability by training exclusively on *safe* inputs.

Formulation as anomaly detection. To train using only safe inputs, we formulate the task as anomaly detection. During training, the module learns common patterns in safe inputs. During testing, deviations from the learned safety pattern indicate attacks or unsafe inputs. To this end, we introduce the Safety Pattern Auto-Encoder (SPAE), which is trained to reconstruct MSCAVs of safe inputs. During testing, a low reconstruction error indicates in-distribution safe inputs, while a high reconstruction error indicates possible attacks or unsafe inputs, following existing works in anomaly detection (Sakurada and Yairi, 2014). We next describe the architecture of SPAE, as well as its training and testing strategies in detail.

SPAE architecture. SPAE uses a standard auto-encoder with a three-layer fully connected encoder and a symmetric decoder with ReLU activations. This hierarchical non-linear structure enables the model to capture complex inter-layer safety relations, considering that each dimension in SCAVs correspond to one layer. Formally, the encoder f_{enc} compresses the MSCAV $\mathbf{Sr}$ into a low-dimensional latent space, and the decoder f_{dec} reconstructs it back to the original dimensionality $|\mathcal{L}\text{s}|$:

$$\hat{\mathbf{Sr}} = f_{\text{dec}}\big(f_{\text{enc}}(\mathbf{Sr})\big). \tag{5}$$

SPAE training. The model is trained solely on safe inputs to minimize assumptions about attacks. The training objective minimizes the mean squared reconstruction error between the input and reconstructed MSCAVs:

$$\mathcal{L}_{\text{rec}} = \frac{1}{N_s}\sum_{i=1}^{N_s} ||\hat{\mathbf{Sr}} - \mathbf{Sr}||_2^2. \tag{6}$$

Here, N_s is the number of safe inputs. This loss encourages the auto-encoder to learn a compact representation of safety patterns.

SPAE testing. During testing, SPAE detects anomalies by evaluating the reconstruction error:

$$\delta = ||\hat{\mathbf{Sr}} - \mathbf{Sr}||_2^2. \tag{7}$$

Given an input (safe or attacked), if δ exceeds a threshold τ, the input is flagged as attacked. Otherwise, the input is considered safe. Threshold selection is detailed in Appendix B.2.

Advantages over prior methods. Unlike approaches that rely solely on cosine similarity over raw LVLM embeddings (Jiang et al., 2025) or selected gradients (Xie et al., 2024), which assume feature independence and equal contribution of features, SPAE explicitly learns inter-feature dependencies of safety patterns. This design eliminates strong independence assumptions and substantially improves detection accuracy (Sec. 3.2).

3 Experiments

3.1 Experimental Setups

Jailbreak attack methods. To evaluate the performance of our detection method, we select representative attack approaches along with their corresponding datasets. For *prompt-manipulation-based* attacks, we use: **MML** (Wang et al., 2024b), **JOOD** (Jeong et al., 2025), and **HADES** (Li et al., 2025). For *adversarial-perturbation-based* attacks, we employ **VAJM** (Qi et al., 2024) and **UMK** (Wang et al., 2024a). Details of these attack methods are introduced in Appendix D.

Training and validation datasets. To train LoD and select hyperparameters, we use unsafe inputs $\mathbf{I}^-$ from AdvBench (Chen et al., 2022), which contains prompts describing unsafe behaviors. Safe inputs $\mathbf{I}^+$ are from GQA (Hudson and Manning, 2019), a large-scale visual reasoning dataset for compositional question answering. We combine prompts from both datasets and create multimodal inputs by synthesizing images for both unsafe and safe texts using Pixart-Sigma (Chen et al., 2024). Among the resulting safe-unsafe pairs, 100 are used

Model	Method	Results on Jailbreak Attacks (AUROC)					Min	Average
		MML	JOOD	HADES	VAJM	UMK		
LLaVA	MirrorCheck$^{(a)}$	0.5113	0.4993	0.4420	0.9325	0.3723	0.3723	0.5515
	CIDER$^{(a)}$	0.2409	0.7403	0.5983	0.3489	0.3258	0.2409	0.4508
	JailGuard$^{(a)}$	0.6159	0.7919	0.7813	0.7234	0.7335	0.6159	0.7292
	GradSafe$^{(b)}$	**1.0000**	0.8584	0.9690	0.7680	0.0563	0.0563	0.7303
	HiddenDetect$^{(b)}$	0.4527	0.9577	0.9975	0.9932	0.9312	0.4527	0.8665
	Ours$^{(b)}$	**1.0000**	**0.9997**	**1.0000**	**1.0000**	**1.0000**	**0.9997**	**0.9999**
	Δ	+0%	+4.38%	+0.25%	+0.68%	+7.38%	+62.31%	+15.04%
Qwen2.5-VL	MirrorCheck$^{(a)}$	0.7118	0.5531	0.5352	0.3619	0.2683	0.2683	0.4897
	CIDER$^{(a)}$	0.2537	0.7395	0.7402	0.8441	0.5016	0.2537	0.6158
	JailGuard$^{(a)}$	0.2664	0.7000	0.6061	0.5355	0.5222	0.2664	0.5260
	GradSafe$^{(b)}$	0.9948	0.9149	0.9030	0.9867	0.9660	0.9030	0.9531
	HiddenDetect$^{(b)}$	0.3369	0.8507	0.9154	0.7440	0.5327	0.3369	0.6759
	Ours$^{(b)}$	**1.0000**	**0.9999**	**1.0000**	**1.0000**	**1.0000**	**0.9999**	**0.9999**
	Δ	+0.52%	+9.29%	+9.24%	+1.34%	+3.52%	+10.73%	+4.91%
CogVLM	MirrorCheck$^{(a)}$	0.5974	0.5229	0.4177	0.6193	0.7323	0.4177	0.5778
	CIDER$^{(a)}$	0.2066	0.7636	0.5646	0.4269	0.1305	0.1305	0.4184
	JailGuard$^{(a)}$	0.5279	0.7658	0.6720	0.9020	0.8186	0.5279	0.7372
	GradSafe$^{(b)}$	**1.0000**	0.4520	0.5424	0.9036	0.8754	0.4520	0.7547
	HiddenDetect$^{(b)}$	0.9132	0.8859	0.9635	0.9769	0.9677	0.8859	0.9414
	Ours$^{(b)}$	**1.0000**	**0.9998**	**1.0000**	**1.0000**	**1.0000**	**0.9998**	**0.9999**
	Δ	+0%	+12.86%	+3.78%	+2.36%	+3.34%	+12.86%	+6.21%

Table 1: Comparison of detection AUROC across different LVLMs and attack methods. Methods are categorized into two categories based on whether they utilize only the input and output of LVLMs$^{(a)}$ or utilize information of internal activations or gradients in the LVLMs$^{(b)}$. The best results are shown in **bold**, and the second-best results are underlined. Δ denotes the relative improvement of our LoD method over the best baseline.

for training the MSCAV classifiers, and 100 pairs are used for validation. The autoencoder is trained on 320 safe inputs, reserving 80 for validation.

Testing datasets. We use two sets of testing data. The *first set* includes 400 safe inputs from SEED-Bench (Li et al., 2024) and 400 unsafe inputs from MM-SafetyBench (Liu et al., 2024c). The attacked inputs are created by applying the attack methods on the unsafe inputs, except for HADES, which provides its own dataset (Li et al., 2025) that we directly use. In the *second set*, the safe inputs are replaced by 400 samples from MM-Vet2. We report the results of the first set in the main paper and the results of the second set in Appendix A.

Baseline. To ensure a comprehensive evaluation and highlight the strengths of our proposed method, we select representative baselines from two main categories of detection approaches: (a) For methods that do not use internal representations, we include CIDER (Xu et al., 2024a), MirrorCheck (Fares et al., 2024), and JailGuard (Zhang et al., 2025); (b) For methods leveraging internal representations, we incorporate HiddenDetect (Jiang et al., 2025) and GradSafe (Xie et al., 2024). For GradSafe, we use GradSafe-Zero instead of GradSafe-Adapt, as the latter requires domain adaptation for jailbreak inputs, whereas our goal is to detect attacks without jailbreak labels. Results for GradSafe-Adapt are reported in Appendix B.3, showing limited generalization to unseen attacks.

Evaluation criteria. We follow Jiang et al. (2025) and evaluate detection accuracy using **AUROC**, which reflects performance across all decision thresholds. Results on additional criteria, including **F1** and TPR at low FPR (e.g., FPR ≤ 0.01), are reported in Appendix B. We also report the computation time in **seconds** to analyze efficiency.

LVLMs. We evaluate our method on three LVLMs: LLaVA-1.6-vicuna-7B (Liu et al., 2024b), CogVLM-chat-v1.1 (Wang et al., 2023), and Qwen2.5-VL-7B-Instruct (Bai et al., 2025).

Hyperparameter settings. We set the hyperparameters as follows for all experiments. For P_0, we choose 0.9. Although AUROC is independent of the specific threshold, we set the thresholds for the three LVLMs as described in Appendix B.2. The settings for the baselines are given in Appendix E.

3.2 Overall Detection Performance

Table 1 presents the AUROC scores of various detection methods across three LVLMs and five jailbreak attack methods. Here, we conduct the experiments on the first set of testing data (Section 3.1). The results on the second set of testing data are given in Appendix A. Based on the results, we have two major observations:

First, our proposed LoD consistently achieves the highest performance, outperforming the strongest baselines with relative improvements ranging from 4.91% to 15.04%, and substantially improving worst-case performance (i.e., minimum AUROC across attacks) with gains up to 62.31% on LLaVA, 10.73% on Qwen2.5-VL, and 12.86% on CogVLM, demonstrating strong detection performance across diverse unseen attacks. We analyze why our method achieves a high AUROC of around 0.999 in Appendix C.

Second, baselines that leverage internal representations (GradSafe and HiddenDetect), generally achieve higher average AUROC than non-internal methods (MirrorCheck, CIDER, and JailGuard), reflecting the benefits of utilizing internal representations. However, they do not always achieve better minimum AUROC (see the **Min** column), indicating that their heuristic principles may not be universally effective across all attacks and LVLMs.

3.3 Ablation Study

To better understand the contribution of each core component in the LoD framework, we compare our method with the following ablated variants:

- **LoD w/o MSCAV**: This variant assesses the effectiveness of the representation learning module by replacing MSCAVs with raw internal activations concatenated across all layers.
- **LoD w/o SPAE**: This variant demonstrates the effectiveness of SPAE by replacing it with a simple aggregation of MSCAV features: an input is flagged as unsafe if any dimension of the refined MSCAV $\mathbf{S}_r$ exceeds its corresponding threshold selected using the validation set (Appendix B.2).
- **LoD w/ HiddenDetect**: This variant compares our MSCAV-based representation learning with that in an existing state-of-the-art method, HiddenDetect. Specifically, we replace our MSCAVs with the feature extraction method proposed in HiddenDetect (Jiang et al., 2025).

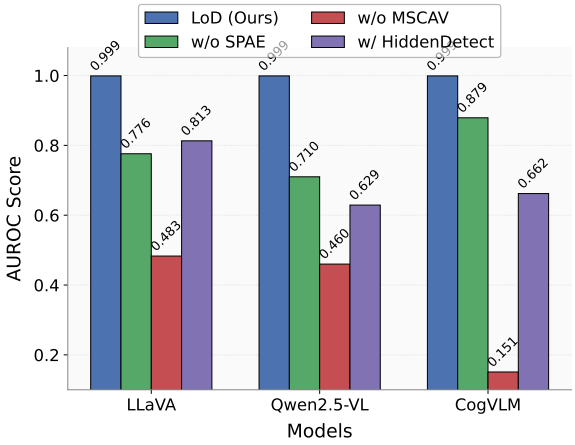


Figure 5: Ablation study results showing the average AUROC scores on three LVLMs across five attack methods under different ablation settings. Both MSCAV and SPAE contribute to the detection accuracy.

The ablation study results in Figure 5 highlight the necessity of both MSCAV and SPAE in the LoD framework. Removing the MSCAV module (w/o MSCAV) leads to the most severe degradation across all three LVLMs (AUROC from 0.151 to 0.483), underscoring its critical role in extracting discriminative safety-relevant features and mitigating redundancy in raw activations. When replacing MSCAV with HiddenDetect features (w/ HiddenDetect), performance improves relative to the ablated variants but remains notably below the full LoD model (AUROC from 0.629 to 0.813), indicating that LoD's MSCAV better generalizes safety signals across diverse LVLMs. Similarly, removing SPAE (w/o SPAE) also causes consistent drops

P_0	Average AUROC over five attacks		
	LLaVA	**Qwen2.5-VL**	**CogVLM**
0.80	0.9999	0.9999	0.9999
0.90	0.9999	0.9999	0.9999
0.95	0.9998	0.9999	0.9999
0.97	0.9999	0.9999	0.9999
0.99	0.9993	0.9999	0.9998

Table 2: Average AUROC scores over five attack methods for different P_0 thresholds across three LVLMs. The threshold we used is highlighted in gray. An analysis of why our method achieves a high AUROC of approximately 0.999 is provided in Appendix C.

(AUROC from 0.710 to 0.879), demonstrating that modeling inter-layer interactions by using SPAE is essential for improving detection performance.

3.4 Parameter Sensitivity Analysis

Our method involves two hyperparameters: the layer selection threshold P_0 and the decision threshold τ. Because AUROC is computed by integrating over all possible τ values, it is not affected by choices of τ. We therefore focus on analyzing the impact of P_0 here, while the effect of τ on threshold-dependent criteria (e.g., F1-score) is discussed in Appendix B.2.

Table 2 shows that the average AUROC remains consistently high for all LVLMs and P_0 values, indicating that LoD is highly robust to the layer selection threshold. Moderate P_0 values (e.g., 0.90) yield uniformly strong results, while overly large P_0 (e.g., 0.99) may cause minor degradation on LLaVA and Cogvlm due to the removal of informative layers. These results confirm that LoD maintains stable detection performance over a wide hyperparameter range.

3.5 Computational Efficiency Analysis

As shown in Table 3, our LoD method is the most efficient across three LVLMs, with a computation time ranging from 0.25 to 0.62 seconds. Here, most methods are run on a single NVIDIA A800 GPU, except for GradSafe on Qwen2.5-VL, which is run on 4 GPUs due to high memory usage. As shown in the table, LoD reduces detection time by up to 62.7% compared to the best baseline on LLaVA, and shows improvements of 21.0% and 12.8% on Qwen2.5-VL and CogVLM, respectively. In terms of training cost, fitting our MSCAV classifiers and the SPAE module requires only 12 and 2 seconds, respectively, which is a one-time offline training. These results highlight the efficiency of LoD.

Method	LLaVA	Qwen2.5-VL	CogVLM
MirrorCheck	1.99	2.16	3.08
CIDER	0.91	<u>0.75</u>	1.38
JailGuard	15.22	21.24	16.39
GradSafe	0.85	0.87†	1.08
HiddenDetect	<u>0.67</u>	0.86	<u>0.39</u>
Ours	**0.25**	**0.62**	**0.34**
Δ	+62.7%	+21.0%	+12.8%

Table 3: Average detection time (in seconds) per input across three LVLMs. Best and second best results are shown in **bold** and <u>underlined</u>, respectively. Δ denotes the relative improvement of LoD over the best baseline.

4 Related Work

Detection methods. Existing detection methods can be categorized into two groups. Learning-based methods are typically trained on specific attacked data (Alon and Kamfonas, 2023; Pi et al., 2024; Team, 2024; Xie et al., 2024). Although they achieve high accuracy on known attacks, their ability to generalize to unseen attacks is not guaranteed. In contrast, non-learning-based methods (Xu et al., 2024a; Fares et al., 2024; Xie et al., 2024; Jiang et al., 2025) remove the dependence on attack-specific training data by relying on heuristic principles and can be applied to unseen attacks. However, their performance remains limited in complex or unforeseen scenarios. Unlike these methods, our approach trains a learnable framework on both safe and unsafe inputs without attack-specific data, ensuring strong generalization and high detection performance.

Jailbreak attacks. The multimodal nature of LVLMs creates new attack surfaces, especially in the visual modality, increasing vulnerability to jailbreaks (Fan et al., 2024; Jin et al., 2024; Liu et al., 2024d,a). Existing attacks fall into two categories (Liu et al., 2024a): prompt manipulation and adversarial perturbation. Prompt manipulation alters textual or visual inputs to bypass safety filters, usually without gradients (Jeong et al., 2025; Wang et al., 2024b; Gong et al., 2023; Ma et al., 2024; Liu et al., 2024c; Li et al., 2025), while adversarial perturbation uses gradient-based changes to induce unsafe outputs (Niu et al., 2024; Qi et al., 2024; Wang et al., 2024a; Shayegani et al., 2023). Although existing detection approaches often fail to generalize or maintain satisfactory performance across different attacks, our learnable framework can accurately detect unseen jailbreaks.

5 Conclusion

In this work, we introduce Learning to Detect, a learnable framework that leverages LVLMs' internal activations for jailbreak detection. While existing methods often struggle to achieve both high accuracy and generalization, LoD solves this by training task-specific parameters on safe and unsafe data, without relying on attack-specific data. Experiments show that LoD consistently outperforms state-of-the-art baselines across multiple LVLMs and previously unknown attacks.

Limitations

Although our method achieves consistent performance across models and attacks, it has several limitations. First, its results depend on the specific LVLM used. Major architectural changes or new safety alignment mechanisms may require adapting the detector, though the prevalence of deep neural architectures suggests our approach will remain broadly applicable. Second, while our method is demonstrated to be effective against diverse jailbreaks, our evaluation cannot fully anticipate future adaptive attacks that specifically manipulate internal representations. Exploring such scenarios is an important step toward building more resilient defenses.

References

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. 2024. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *European Conference on Computer Vision*, pages 74–91. Springer.

Yangyi Chen, Hongcheng Gao, Ganqu Cui, Fanchao Qi, Longtao Huang, Zhiyuan Liu, and Maosong Sun. 2022. Why should adversarial perturbations be imperceptible? rethink the research paradigm in adversarial nlp. *arXiv preprint arXiv:2210.10683*.

Giovanni De Gasperis and Sante Dino Facchini. 2025. A comparative study of rule-based and data-driven approaches in industrial monitoring. *arXiv preprint arXiv:2509.15848*.

Yihe Fan, Yuxin Cao, Ziyu Zhao, Ziyao Liu, and Shaofeng Li. 2024. Unbridled icarus: A survey of the potential perils of image inputs in multimodal large language model security.

Samar Fares, Klea Ziu, Toluwani Aremu, Nikita Durasov, Martin Takáč, Pascal Fua, Karthik Nandakumar, and Ivan Laptev. 2024. Mirrorcheck: Efficient adversarial defense for vision-language models. *arXiv preprint arXiv:2406.09250*.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. 2023. Figstep: Jailbreaking large vision-language models via typographic visual prompts. *arXiv preprint arXiv:2311.05608*.

Tianle Gu, Zeyang Zhou, Kexin Huang, Dandan Liang, Yixu Wang, Haiquan Zhao, Yuanqi Yao, Xingge Qiao, Keqing Wang, and Yujiu Yang. 2024. Mllmguard: A multi-dimensional safety evaluation suite for multimodal large language models.

Drew A. Hudson and Christopher D. Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *IEEE*.

Joonhyun Jeong, Seyun Bae, Yeonsung Jung, Jaeryong Hwang, and Eunho Yang. 2025. Playing the fool: Jailbreaking llms and multimodal llms with out-of-distribution strategy. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29937–29946.

Yilei Jiang, Xinyan Gao, Tianshuo Peng, Yingshui Tan, Xiaoyong Zhu, Bo Zheng, and Xiangyu Yue. 2025. Hiddendetect: Detecting jailbreak attacks against large vision-language models via monitoring hidden states. *arXiv preprint arXiv:2502.14744*.

Haibo Jin, Leyang Hu, Xinuo Li, Peiyan Zhang, Chonghan Chen, Jun Zhuang, and Haohan Wang. 2024. Jailbreakzoo: Survey, landscapes, and horizons in jailbreaking large language and vision-language models. *arXiv preprint arXiv:2407.01599*.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. 2024. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13299–13308.

Yifan Li, Hangyu Guo, Kun Zhou, Wayne Xin Zhao, and Ji Rong Wen. 2025. Images are achilles' heel ofalignment: Exploiting visual vulnerabilities forjailbreaking multimodal large language models. In *European Conference on Computer Vision*.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. 2023. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. *arXiv preprint arXiv:2310.17389*.

Daizong Liu, Mingyu Yang, Xiaoye Qu, Pan Zhou, Yu Cheng, and Wei Hu. 2024a. A survey of attacks on large vision-language models: Resources, advances, and future trends. *arXiv preprint arXiv:2407.07403*.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. Llava-next: Improved reasoning, ocr, and world knowledge.

Xin Liu, Yichen Zhu, Jindong Gu, Yunshi Lan, Chao Yang, and Yu Qiao. 2024c. Mm-safetybench: A benchmark for safety evaluation of multimodal large language models. In *European Conference on Computer Vision*, pages 386–403. Springer.

Xin Liu, Yichen Zhu, Yunshi Lan, Chao Yang, and Yu Qiao. 2024d. Safety of multimodal large language models on images and texts. *arXiv preprint arXiv:2402.00357*.

Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. 2024. Jailbreakv: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks. *arXiv preprint arXiv:2404.03027*.

Siyuan Ma, Weidi Luo, Yu Wang, and Xiaogeng Liu. 2024. Visual-roleplay: Universal jailbreak attack on multimodal large language models via role-playing image character. *arXiv preprint arXiv:2405.20773*.

Zhenxing Niu, Haodong Ren, Xinbo Gao, Gang Hua, and Rong Jin. 2024. Jailbreaking attack against multimodal large language model. *arXiv preprint arXiv:2402.02309*.

Renjie Pi, Tianyang Han, Jianshu Zhang, Yueqi Xie, Rui Pan, Qing Lian, Hanze Dong, Jipeng Zhang, and Tong Zhang. 2024. Mllm-protector: Ensuring mllm's safety without hurting performance. *arXiv preprint arXiv:2401.02906*.

Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. 2024. Visual adversarial examples jailbreak aligned large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 21527–21536.

Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. *ACM*.

Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. 2023. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models.

Llama Team. 2024. Meta llama guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.

Ruofan Wang, Xingjun Ma, Hanxu Zhou, Chuanjun Ji, Guangnan Ye, and Yu-Gang Jiang. 2024a. White-box multimodal jailbreaks against large vision-language models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6920–6928.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. 2023. Cogvlm: Visual expert for pretrained language models.

Yu Wang, Xiaofei Zhou, Yichen Wang, Geyuan Zhang, and Tianxing He. 2024b. Jailbreak large vision-language models through multi-modal linkage. *arXiv preprint arXiv:2412.00473*.

Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. Gradsafe: Detecting jailbreak prompts for llms via safety-critical gradient analysis. *arXiv preprint arXiv:2402.13494*.

Yue Xu, Xiuyuan Qi, Zhan Qin, and Wenjie Wang. 2024a. Cross-modality information check for detecting jailbreaking in multimodal large language models. *arXiv preprint arXiv:2407.21659*.

Zhihao Xu, Ruixuan Huang, Changyu Chen, and Xiting Wang. 2024b. Uncovering safety risks of large language models through concept activation vector. *Advances in Neural Information Processing Systems*, 37:116743–116782.

Mang Ye, Xuankun Rong, Wenke Huang, Bo Du, Nenghai Yu, and Dacheng Tao. 2025a. A survey of safety on large vision-language models: Attacks, defenses and evaluations. *arXiv preprint arXiv:2502.14881*.

Zhoutong Ye, Mingze Sun, Huan-ang Gao, Chun Yu, and Yuanchun Shi. 2025b. Moat: Evaluating lmms for capability integration and instruction grounding. *arXiv preprint arXiv:2503.09348*.

Weihao Yu, Zhengyuan Yang, Lingfeng Ren, Linjie Li, Jianfeng Wang, Kevin Lin, Chung-Ching Lin, Zicheng Liu, Lijuan Wang, and Xinchao Wang. 2024. Mm-vet v2: A challenging benchmark to evaluate large multimodal models for integrated capabilities. *arXiv preprint arXiv:2408.00765*.

Duzhen Zhang, Yahan Yu, Jiahua Dong, Chenxing Li, Dan Su, Chenhui Chu, and Dong Yu. 2024. Mm-llms: Recent advances in multimodal large language models.

Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. 2025. Jailguard: A universal detection framework for prompt-based attacks on llm systems. *ACM Transactions on Software Engineering and Methodology*.

A Supplementary Evaluation with MM-Vet2

In this section, we provide supplementary evaluation results using the second set of testing data, in which the safe inputs are from MM-Vet2 (Yu et al., 2024). This further validates the accuracy and generalization of our proposed LoD method.

As shown in Table 4, across all three models, LoD maintains high accuracy and strong generalization when evaluated with MM-Vet2 as the safe test dataset. Specifically, LoD achieves the highest average AUROC of 0.9798 on LLaVA, 0.9820 on Qwen2.5-VL, and 0.9677 on CogVLM, outperforming the strongest baselines by 21.49%, 4.51%, and 18.07%, respectively. It also obtains the best minimum AUROC in most cases, indicating enhanced generalization against diverse and challenging attacks. While GradSafe slightly outperforms LoD on a few individual attack types (e.g., MML

Model	Method	Results on Jailbreak Attacks (AUROC)					Min	Average
		MML	JOOD	HADES	VAJM	UMK		
LLaVA	MirrorCheck$^{(a)}$	0.5120	0.4958	0.4408	0.9200	0.3598	0.3598	0.5457
	CIDER$^{(a)}$	0.1966	0.6673	0.5293	0.2984	0.2761	0.1966	0.3935
	JailGuard$^{(a)}$	0.3485	0.5417	0.5507	0.4903	0.4728	0.3485	0.4808
	GradSafe$^{(b)}$	**0.9963**	0.6651	0.8125	0.5371	0.0327	0.0327	0.6079
	HiddenDetect$^{(b)}$	0.3253	0.9090	0.9899	0.9700	0.8383	0.3253	0.8065
	Ours$^{(b)}$	0.9706	**0.9636**	**0.9989**	**0.9958**	**0.9703**	**0.9636**	**0.9798**
	Δ	-2.57%	+6.00%	+0.90%	+2.66%	+15.75%	+176.50%	+21.49%
Qwen2.5-VL	MirrorCheck$^{(a)}$	0.6454	0.4990	0.4797	0.3257	0.2447	0.2447	0.4389
	CIDER$^{(a)}$	0.1546	0.6175	0.5961	0.7112	0.3719	0.2537	0.4903
	JailGuard$^{(a)}$	0.2072	0.5367	0.4965	0.4356	0.4140	0.2072	0.4180
	GradSafe$^{(b)}$	**0.9857**	0.8881	0.8892	0.9790	0.9427	0.8881	0.9396
	HiddenDetect$^{(b)}$	0.5763	0.9407	0.9671	0.8642	0.7315	0.5763	0.8160
	Ours$^{(b)}$	0.9355	**0.9877**	**0.9991**	**0.9997**	**0.9882**	**0.9355**	**0.9820**
	Δ	-5.09%	+5.00%	+3.31%	+2.11%	+4.83%	+5.34%	+4.51%
CogVLM	MirrorCheck$^{(a)}$	0.5915	0.5228	0.4175	0.6115	0.7367	0.4175	0.5760
	CIDER$^{(a)}$	0.2320	0.7282	0.5462	0.4199	0.1433	0.1433	0.4139
	JailGuard$^{(a)}$	0.3835	0.6087	0.5266	0.7933	0.6871	0.3835	0.5998
	GradSafe$^{(b)}$	**1.0000**	0.3559	0.4002	0.7693	0.6950	0.3559	0.6369
	HiddenDetect$^{(b)}$	0.6944	0.7558	0.8936	0.8845	0.8695	0.6944	0.8196
	Ours$^{(b)}$	0.8912	**0.9657**	**0.9973**	**0.9939**	**0.9903**	**0.8912**	**0.9677**
	Δ	-10.88%	+27.88%	+11.60%	+12.37%	+13.89%	+28.34%	+18.07%

Table 4: Comparison of detection AUROC across different LVLMs and attack methods. The experiment is conducted on the second test set, in which the safe inputs are from MM-Vet2. Methods are categorized into two categories based on whether they utilize only the input and output information of LVLMs$^{(a)}$ or utilize information of internal representations in the LVLMs$^{(b)}$. The best results are shown in **bold**, and the second-best results are underlined. Δ denotes the relative improvement of our LoD method over the best baseline.

on LLaVA), LoD still delivers the best overall detection performance across models and attacks. These results further confirm that LoD achieves both high accuracy and strong generalization without relying on attack-specific data. Furthermore, we analyze why we stably obtain high AUROC scores in Appendix C.

B Supplementary Evaluation Using Additional Criteria

To further validate the effectiveness and robustness of our detection method, we note that the main text already demonstrates high AUROC scores. However, AUROC is an average criterion that may not fully reflect performance under specific operating conditions. In this section, we pursue two objectives: first, we measure TPR at low FPR to evaluate detection capability under strict false positive constraints; second, we determine practical thresholds for our method and verify its effectiveness at these thresholds.

B.1 TPR at Low FPR

Table 5 reports the TPR at a fixed FPR of 0.01. Most baseline methods exhibit very low TPR, often close to zero. This is because such a stringent FPR threshold only accepts the most confident predictions as positives; many jailbreak inputs are scored too similarly to safe inputs, causing the detectors to miss almost all attacks. GradSafe and HiddenDetect achieve moderate TPRs, suggesting that internal-representation-based methods can capture certain attack representations, but their effectiveness is highly sensitive to both the model and the jailbreak methods.

In contrast, our method consistently achieves competitive TPR across all attacks and LVLMs, reliably detecting true positives even under this strict operating point. These results highlight the accuracy and generalization of LoD, demonstrating its ability to generalize across diverse jailbreak strategies.

Model	Method	Results on Jailbreak Attacks (TPR @ FPR=0.01)					Average
		MML	JOOD	HADES	VAJM	UMK	
LLaVA	MirrorCheck$^{(a)}$	0.000	0.010	0.000	0.083	0.000	0.018
	CIDER$^{(a)}$	0.000	0.087	0.033	0.000	0.000	0.024
	JailGuard$^{(a)}$	0.018	0.130	0.193	0.140	0.070	0.110
	GradSafe$^{(b)}$	**1.000**	0.593	0.868	0.380	0.000	0.568
	HiddenDetect$^{(b)}$	0.000	0.580	0.953	0.802	0.120	0.491
	Ours$^{(b)}$	**1.000**	**0.998**	**1.000**	**1.000**	**1.000**	**0.999**
	Δ	+0.0%	+68.3%	+4.9%	+24.7%	+733.3%	+75.9%
Qwen2.5-VL	MirrorCheck$^{(a)}$	0.015	0.018	0.005	0.003	0.000	0.008
	CIDER$^{(a)}$	0.000	0.133	0.028	0.043	0.005	0.042
	JailGuard$^{(a)}$	0.000	0.005	0.013	0.022	0.038	0.015
	GradSafe$^{(b)}$	0.998	0.715	0.782	0.922	0.810	0.846
	HiddenDetect$^{(b)}$	0.000	0.083	0.205	0.148	0.000	0.087
	Ours$^{(b)}$	**1.000**	**0.998**	**1.000**	**1.000**	**1.000**	**0.999**
	Δ	+0.2%	+39.6%	+27.9%	+8.5%	+23.5%	+18.1%
CogVLM	MirrorCheck$^{(a)}$	0.003	0.007	0.000	0.007	0.000	0.004
	CIDER$^{(a)}$	0.000	0.065	0.000	0.000	0.000	0.013
	JailGuard$^{(a)}$	0.003	0.160	0.163	0.440	0.283	0.209
	GradSafe$^{(b)}$	**1.000**	0.133	0.113	0.568	0.243	0.411
	HiddenDetect$^{(b)}$	0.000	0.403	0.770	0.695	0.628	0.499
	Ours$^{(b)}$	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Δ	+0.0%	+148.1%	+29.9%	+43.9%	+59.2%	+100.4%

Table 5: TPR at FPR=0.01 across different LVLMs and detection methods. Methods are categorized into two categories based on whether they utilize only the input and output information of LVLMs$^{(a)}$ or utilize information of internal representations in the LVLMs$^{(b)}$. The safe-test data are sampled from **SEED-Bench**. The best results are shown in **bold**, and the second-best results are shown in underline. Δ denotes the relative improvement of our method over the best baseline.

B.2 Threshold Decision

To evaluate the effect of the SPAE, we set the threshold for each dimension as the maximum value of that MSCAV dimension on the auto-encoder's validation set of safe inputs, with the results shown in Figure 5.

To determine thresholds for applying our method in real-world scenarios, we use the MOAT (Ye et al., 2025b) dataset, which contains approximately 200 items, each consisting of a question for VLMs paired with a corresponding image. MOAT serves as a collection of safe samples, which we use to align the measurement scores and select the 90th percentile as the threshold, thereby reducing the false positive rate. The resulting thresholds are 0.07502 for LLaVA, 0.07148 for Qwen2.5-VL, and 0.04083 for CogVLM. We apply the same procedure to determine thresholds for all baselines.

To evaluate the effectiveness of the chosen threshold, we report F1 scores across models and datasets in Table 5. Our method consistently achieves high values, significantly outperforming all baselines. For Vicuna, we improve over the best baseline by +42.1% on average, with especially large gains on UMK (+46.8%). Similar improvements hold for Qwen2.5-VL (+7.7% on average) and CogVLM (+57.7%), indicating that the gains are not tied to a single model but are robust across architectures. Notably, while baselines such as GradSafe or HiddenDetect achieve strong performance on certain datasets but collapse on others (often dropping to near zero), our method maintains consistently high F1 across all settings. Such inconsistencies suggest that thresholds determined by these baselines lack generalizability: their score measurements for

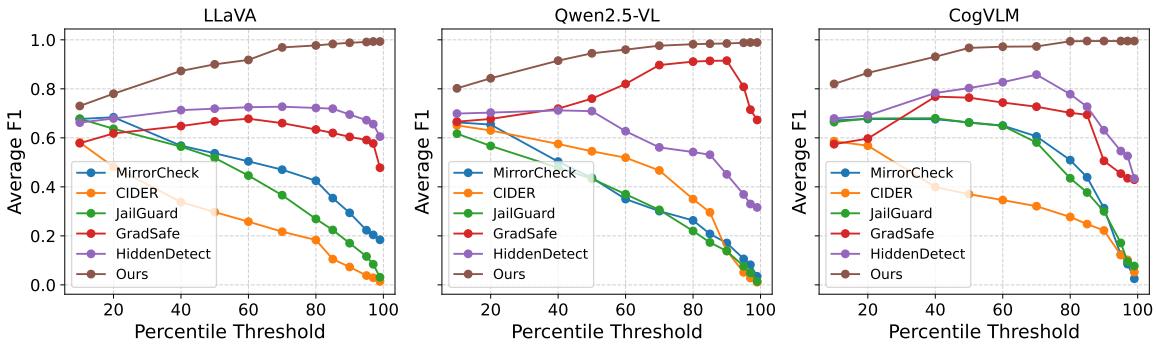


Figure 6: F1 score sensitivity of different detection methods on the three models across threshold percentiles from 10th to 99th. Our method consistently achieves the highest F1 at each percentile, demonstrating robustness to threshold selection.

Model	Method	Results on Jailbreak Attacks (F1)					Average
		MML	JOOD	HADES	VAJM	UMK	
LLaVA	MirrorCheck$^{(a)}$	0.117	0.272	0.080	0.896	0.106	0.294
	CIDER$^{(a)}$	0.000	0.274	0.093	0.000	0.000	0.073
	JailGuard$^{(a)}$	0.034	0.209	0.267	0.221	0.117	0.170
	GradSafe$^{(b)}$	**0.998**	0.695	0.902	0.427	0.000	0.604
	HiddenDetect$^{(b)}$	0.000	0.869	0.968	0.967	0.673	0.695
	Ours$^{(b)}$	0.988	**0.988**	**0.988**	**0.988**	**0.988**	**0.988**
	Δ	-1.0%	+13.7%	+2.1%	+2.2%	+46.8%	+42.1%
Qwen2.5-VL	MirrorCheck$^{(a)}$	0.420	0.251	0.157	0.026	0.000	0.171
	CIDER$^{(a)}$	0.000	0.321	0.098	0.253	0.024	0.139
	JailGuard$^{(a)}$	0.000	0.078	0.247	0.209	0.158	0.138
	GradSafe$^{(b)}$	**0.985**	0.846	0.879	0.957	0.910	0.915
	HiddenDetect$^{(b)}$	0.000	0.718	0.824	0.592	0.120	0.451
	Ours$^{(b)}$	**0.985**	**0.985**	**0.985**	**0.985**	**0.985**	**0.985**
	Δ	+0.0%	+16.4%	+12.1%	+2.9%	+8.2%	+7.7%
CogVLM	MirrorCheck$^{(a)}$	0.321	0.283	0.065	0.348	0.546	0.313
	CIDER$^{(a)}$	0.000	0.639	0.263	0.206	0.000	0.222
	JailGuard$^{(a)}$	0.000	0.247	0.255	0.590	0.406	0.300
	GradSafe$^{(b)}$	**0.995**	0.232	0.200	0.717	0.387	0.506
	HiddenDetect$^{(b)}$	0.000	0.608	0.876	0.849	0.821	0.631
	Ours$^{(b)}$	**0.995**	**0.995**	**0.995**	**0.995**	**0.995**	**0.995**
	Δ	+0.0%	+55.7%	+13.6%	+17.2%	+22.0%	+57.7%

Table 6: F1 scores across different models and methods. Methods are categorized into two categories based on whether they utilize only the input and output information of LVLMs$^{(a)}$ or utilize information of internal representations in the LVLMs$^{(b)}$. The safe-test data are sampled from **SEED-Bench**. Best results among baselines are shown in **bold**, second-best baseline results are shown in underline. Δ denotes the relative improvement of our method over the best baseline.

different safe samples vary considerably, leading to unstable and limited detection performance.

By contrast, our method maintains consistently high F1 across all settings, without significant degradation on any benchmark. This demonstrates not only that the threshold selected using MOAT generalizes well across different models but also that it remains effective under diverse attack

scenarios.

Sensitivity analysis. To evaluate the impact of percentile selection on detection performance in practice, we investigate how the average F1 score changes from the 10th to the 99th percentile. Figure 6 presents the results for different methods and baselines. While the F1 scores of all methods fluctuate to some extent as the percentile varies, our method consistently achieves the highest detection performance with the smallest overall variation, demonstrating minimal sensitivity to percentile selection. This indicates that in practical scenarios, our detection can generalize against different attacks regardless of the chosen percentile. Furthermore, we analyze in Appendix C why the average F1 score steadily increases and remains nearly constant as the percentile threshold rises.

B.3 Results on GradSafe-Adapt

In this section, we further compare GradSafe-Adapt (Xie et al., 2024) with our LoD method. Following their original setting, we conduct domain adaptation using the official training split of the ToxicChat (Lin et al., 2023) benchmark.

As shown in Table 7, while GradSafe-Adapt achieves reasonable detection performance on certain attack types, its effectiveness drops on some previously unseen jailbreak attacks (e.g., UMK for LLaVA and JOOD for CogVLM), leading to large performance gaps in both minimum and average AUROC. In contrast, our LoD method maintains consistently high performance across all attack types and models, demonstrating strong generalization to unseen jailbreak attacks without relying on attack-specific labels.

C Reconstruction Error Statistics

Here, we analyze the reason for achieving a high AUROC of approximately 0.999 by investigating the reconstruction errors of the safe and attacked inputs. If the reconstruction errors of the safe inputs seldom overlap with those of the attacked ones in the testing data, it confirms that our method well separates these two groups of inputs and thus can achieve an AUROC that is close to 1. Our analysis results, shown in Table 8, confirm this assumption.

More specifically, we present the statistical features of reconstruction errors across different datasets: the threshold selection dataset MOAT (Ye et al., 2025b), the safe test sets SEED-Bench (Li et al., 2024) and MM-Vet2 (Yu et al., 2024), the attack dataset HADES (Li et al., 2025), and the four remaining jailbreak attack datasets constructed from perturbations in MM-SafetyBench (Liu et al., 2024c).

In Table 8, we report the reconstruction error statistics (min, max, average, and std) for LLaVA, Qwen2.5-VL, and CogVLM across different datasets and attack types. We observe that the AUROC scores reported in the main experiments (Table 1) are consistently very high. This is largely because the maximum reconstruction errors of SEED-Bench samples are generally lower than the minimum reconstruction errors of many attack datasets, providing a clear separation for detection. For MM-Vet2 (Table 4), although the reconstruction errors are slightly higher than those on SEED-Bench, the mean errors remain substantially smaller than those on attack datasets. Moreover, the moderately larger standard deviations indicate that most reconstruction errors are still below the mean and smaller than the minimum attack errors, which continues to support high AUROC values. In threshold selection experiments, Figure 6 shows that our F1 score steadily increases with the chosen quantile, unlike other baselines that first increase and then decrease. This is because the reconstruction errors in MOAT, used for threshold selection, are consistently smaller than those of SEED-Bench and attack datasets, ensuring that attack inputs are always correctly identified while safe samples are progressively more accurately classified as the threshold increases.

D Jailbreak Attacks

In this section, we introduce the jailbreak attacks used in our experiments. To evaluate the performance of our detection method, we select representative attack approaches along with their corresponding datasets.

For prompt manipulation-based attacks, we use: (a) MML (Wang et al., 2024b), which employs a cross-modal encryption–decryption procedure to reduce overexposure of malicious content; (b) JOOD (Jeong et al., 2025), which transforms inputs into out-of-distribution forms, thereby limiting the model's ability to detect malicious content; and (c) HADES (Li et al., 2025), which integrates image construction and perturbation to generate jailbreak samples.

For adversarial perturbation-based attacks, we employ: (a) VAJM (Qi et al., 2024), which

Model	Method	Results on Jailbreak Attacks					Min	Average
		MML	JOOD	HADES	VAJM	UMK		
LLaVA	GradSafe-Adapt	**1.0000**	0.8584	0.9690	0.7680	0.0563	0.0563	0.7303
	Ours	**1.0000**	**0.9997**	**1.0000**	**1.0000**	**1.0000**	**0.9997**	**0.9999**
	Δ	+0%	+16.47%	+3.20%	+30.20%	+1676.20%	+1675.67%	+17.07%
Qwen2.5-VL	GradSafe-Adapt	0.9953	0.3572	0.9504	0.9898	0.9779	0.3572	0.8541
	Ours	**1.0000**	**0.9999**	**1.0000**	**1.0000**	**1.0000**	**0.9999**	**0.9999**
	Δ	+0.52%	+9.29%	+9.24%	+1.34%	+3.52%	+10.73%	+4.91%
CogVLM	GradSafe-Adapt	0.9961	0.5436	0.9837	0.9731	0.9626	0.5436	0.8918
	Ours	**1.0000**	**0.9998**	**1.0000**	**1.0000**	**1.0000**	**0.9998**	**0.9999**
	Δ	+0.39%	+83.92%	+1.66%	+2.76%	+3.89%	+83.92%	+12.11%

Table 7: Comparison of detection AUROC between LoD and GradSafe-Adapt across different models and datasets. The safe-test data are sampled from **SEED-Bench**. The best results are shown in **bold**. Δ denotes the relative improvement of our LoD method over the GradSafe-Adapt.

applies gradient-based perturbations to images to induce the model to produce unsafe content; and (b) UMK (Wang et al., 2024a), which uses gradient-based perturbations not only on images but also generates adversarial text prefixes to guide the model towards unsafe outputs.

E Settings for Baseline Methods

In this section, we provide the hyperparameter settings for the baseline methods used in our experiments.

- **MirrorCheck:** When testing on a specific model, the captions of images are generated using that same model.
- **CIDER:** The `pair_mode` is set to `injection`, and the denoiser is set to `diffusion`.
- **JailGuard:** The number of variants is set to 8. When testing on a specific model, the model is used to generate responses to each variant.
- **GradSafe:** The default initial safe set and unsafe set are used for selecting safety-critical parameters.
- **HiddenDetect:** The default refusal token list is used. For the selection of the Most Safety-Aware Layer Range, the following layers are used per model: LLaVA (16–29), Qwen2.5-VL (24), and CogVLM (26–26).

Model	Type	Dataset/Attack	Min	Max	Average	Std
LLaVA	Validation	MOAT	0.0002	0.1018	0.0237	0.0286
	Safe test	SEED-Bench	0.0000	0.2382	0.0095	0.0221
	Safe test	MM-Vet2	0.0000	0.7465	0.1074	0.1376
	Attacked test	MML	0.4426	0.5363	0.4804	0.0141
	Attacked test	JOOD	0.0779	0.9597	0.6021	0.2381
	Attacked test	HADES	0.2453	0.9667	0.8971	0.0852
	Attacked test	VAJM	0.3735	0.9557	0.7767	0.1055
	Attacked test	UMK	0.2053	0.8271	0.5193	0.1387
Qwen2.5-VL	Validation	MOAT	0.0004	0.0938	0.0245	0.0261
	Safe test	SEED-Bench	0.0000	0.2086	0.0083	0.0248
	Safe test	MM-Vet2	0.0000	0.5586	0.1058	0.1344
	Attacked test	MML	0.3354	0.4141	0.3824	0.0099
	Attacked test	JOOD	0.1089	0.9763	0.6421	0.1703
	Attacked test	HADES	0.3486	0.9808	0.8343	0.1161
	Attacked test	VAJM	0.4241	0.9853	0.8742	0.1008
	Attacked test	UMK	0.2634	0.8134	0.6016	0.1207
CogVLM	Validation	MOAT	0.0001	0.0572	0.0128	0.0160
	Safe test	SEED-Bench	0.0000	0.1779	0.0024	0.0118
	Safe test	MM-Vet2	0.0000	0.5548	0.0804	0.1004
	Attacked test	MML	0.1915	0.2558	0.2248	0.0113
	Attacked test	JOOD	0.0590	0.6424	0.4424	0.1426
	Attacked test	HADES	0.1668	0.5950	0.5394	0.0436
	Attacked test	VAJM	0.1550	0.6039	0.5259	0.0829
	Attacked test	UMK	0.1948	0.5190	0.4453	0.0629

Table 8: Reconstruction error statistics (min, max, average, std) across different datasets and attacks for Vicuna, Qwen2.5-VL, and CogVLM. The **Type** indicates the dataset's usage: validation data is used for threshold selection (Sec. B.2), while safe-test and attacked-test datasets are used to evaluate the performance of our method and baselines. The evaluation results based on SEED-Bench are shown in Table 1, and the results based on MM-Vet2 are shown in Table 4.